

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-95772

(43) 公開日 平成8年(1996)4月12日

(51) Int.Cl.<sup>6</sup>

G 0 6 F 9/06

識別記号

庁内整理番号

F I

技術表示箇所

5 4 0 F 7230-5B

審査請求 未請求 請求項の数6 ○L (全 12 頁)

(21) 出願番号 特願平6-227613

(22) 出願日 平成6年(1994)9月22日

(71) 出願人 000003078

株式会社東芝

神奈川県川崎市幸区堀川町72番地

(72) 発明者 平山 秀昭

東京都青梅市末広町2丁目9番地 株式会

社東芝青梅工場内

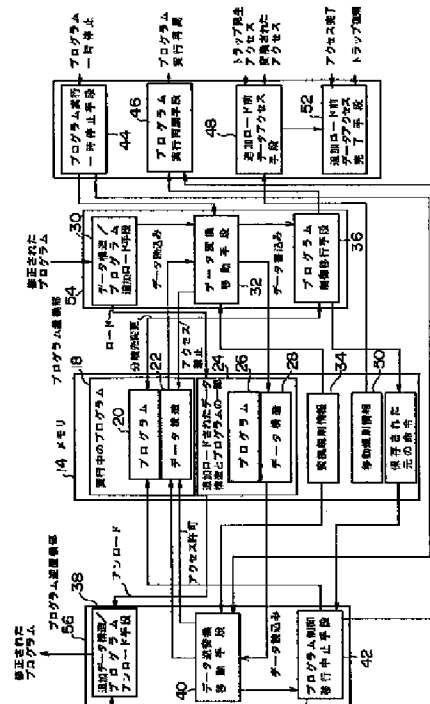
(74) 代理人 弁理士 鈴江 武彦

(54) 【発明の名称】 計算機システム及びプログラム置換方法

(57) 【要約】

【目的】 システムを停止させることなく、実行中のプログラムの部分的な置換を行なう。

【構成】 実行中のプログラムのデータ構造と変更されたプログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段30と、実行中のプログラムを一時的に停止させるプログラム実行一時停止手段44と、停止された実行中のプログラムが使用していたデータ構造から追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段32と、停止された実行中のプログラムを、同プログラムに制御が渡った場合に追加ロードされたプログラムに制御が移るように実行中のプログラムの一部を変更するプログラム制御移行手段36と、データの移動及びプログラムの変更が完了した後、一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段46とを具備する。



## 【特許請求の範囲】

【請求項 1】 計算機システムにおいて、  
 実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段と、  
 前記実行中のプログラムを一時的に停止させるプログラム実行一時停止手段と、  
 前記プログラム実行一時停止手段によって停止された実行中のプログラムが使用していたデータ構造から、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段と、  
 前記プログラム実行一時停止手段によって停止された前記実行中のプログラムを、同プログラムに制御が渡った場合に、前記データ構造／プログラム追加ロード手段によって追加ロードされたプログラムに制御が移るように前記実行中のプログラムの一部を変更するプログラム制御移行手段と、  
 前記データ変換移動手段によるデータの移動、及び前記プログラム制御移行手段によるプログラムの変更が完了した後、前記プログラム実行一時停止手段によって一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段と、  
 を具備したことを特徴とする計算機システム。

【請求項 2】 前記データ構造／プログラム追加ロード手段によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換する追加ロード前データアクセス手段と、  
 前記追加ロード前データアクセス手段による変換によってアクセスが完了した場合に、トラップから復帰させるための追加ロード前データアクセス完了手段と、  
 をさらに具備したことを特徴とする請求項 1 記載の計算機システム。

【請求項 3】 前記データ変換移動手段によってデータが移動された追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させるデータ逆変換移動手段と、  
 前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更するプログラム制御移行中止手段と、  
 前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造とプログラムをアンロードする追加データ構造／プログラムアンロード手段と、  
 をさらに具備したことを特徴とする請求項 1 または請求項 2 記載の計算機システム。

【請求項 4】 計算機システムにおいて、  
 実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードし、

前記実行中のプログラムを一時的に停止し、  
 停止された実行中のプログラムが使用していたデータ構造から、追加ロードされたデータ構造へデータを変換して移動させ、  
 前記実行中のプログラムに制御が渡った場合に、前記追加ロードされたプログラムに制御が移るように前記実行中のプログラムの一部を変更し、  
 前記データの移動、及びプログラムの変更が完了した後、一時的に停止されていたプログラムの実行を再開させることを特徴とするプログラム置換方法。

【請求項 5】 前記データの移動、及びプログラムの変更が完了し、一時的に停止されていたプログラムの実行が再開された後、  
 前記実行中のプログラムの一部によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換し、  
 この変換によってアクセスが完了した場合に、トラップから復帰させることにより移動されたデータにアクセスされるようにすることを特徴とする請求項 4 記載のプログラム置換方法。

【請求項 6】 前記データの移動、及びプログラムの変更が完了し、一時的に停止されていたプログラムの実行が再開された後、  
 追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させ、  
 前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更し、  
 前記追加ロードされたデータ構造とプログラムをアンロードすることを特徴とする請求項 4 記載のプログラム置換方法。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明は、ソフトウェア面で計算機システムの信頼性を向上させるための、計算機システムのプログラム置換方法に関するものである。

## 【0002】

【従来の技術】現在の OS（オペレーティングシステム）、DBMS（データベースマネジメントシステム）、トランザクション処理システム、といった大規模ソフトウェアは、様々な処理要求に対応するため、数百万行の規模に達している。

【0003】しかし、現在のソフトウェア開発技術は、数百万行の規模のプログラムから、バグを取り除くために十分な手法を持ち合わせてはいない。そのため、実際の運用中のプログラムでさえ、バグが内在されたまま、実行が継続されている可能性がある。

【0004】通常、このようなバグは適時発見され、修

正すべき内容を示すバグ修正情報が作成される。バグ修正情報は、プログラムに適用され、改めてコンパイル／リンクされ、新しい実行モジュールが作成される。

【0005】従来、バグが修正された新しい実行モジュールは、バグ修正情報適用前の古い実行モジュールの停止を待って、入れ換えが行なわれている。しかしOS、DBMS、トランザクション処理システムといった長時間実行されるシステムでは、実行モジュールの停止の機会が少ないため、実行モジュールの入れ換えの機会もまた少ない。従って、バグ修正情報が作成されたとしても、実際にそのバグ修正情報をプログラムに適用することが困難になっている。

【0006】今後はシステムの無停止稼働の機会が増え、上記の様な問題は益々深刻となることが予想される。フォールトトレラントシステムによって、ハードウェア故障に起因したシステムダウンは減少させることができたとしても、この様なソフトウェアバグに起因したシステムダウンを減少させることは難しく、しかも処理の高度化と、システムの無停止稼働化に伴って、ソフトウェアバグの問題は益々深刻になってきている。

【0007】

【発明が解決しようとする課題】このように従来のソフトウェア開発環境においては、プログラムのバグを発見し、バグ修正情報を作成し、コンパイル／リンクして新しい実行モジュールを作成したとしても、バグ修正情報適用前の古い実行モジュールが停止される機会を待って、実行モジュールを入れ換えなければならなかった。すなわち、バグ修正情報のプログラムに対する適用には、システムの停止を伴い、バグを検出した時点で逐次修正することができなかったために、計算機システムの信頼性の向上を迅速に図ることができないという問題があった。

【0008】本発明は前記のような事情を考慮してなされたもので、システムを停止させることなく、実行中のプログラムの部分的な置換を行なうことが可能なプログラム置換方法を提供することを目的とする。

【0009】

【課題を解決するための手段】本発明は、実行中のプログラムのデータ構造と変更された前記プログラムの一部を追加ロードするデータ構造／プログラム追加ロード手段と、前記実行中のプログラムを一時的に停止させるプログラム実行一時停止手段と、前記プログラム実行一時停止手段によって停止された実行中のプログラムが使用していたデータ構造から、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造へデータを変換して移動させるデータ変換移動手段と、前記プログラム実行一時停止手段によって停止された前記実行中のプログラムを、同プログラムに制御が渡った場合に、前記データ構造／プログラム追加ロード手段によって追加ロードされたプログラムに制御が移るように前

記実行中のプログラムの一部を変更するプログラム制御移行手段と、前記データ変換移動手段によるデータの移動、及び前記プログラム制御移行手段によるプログラムの変更が完了した後、前記プログラム実行一時停止手段によって一時的に停止されていたプログラムの実行を再開させるプログラム実行再開手段とを具備したことを特徴とする。

【0010】また、前記データ構造／プログラム追加ロード手段によって追加ロードされる前のデータ構造にアクセスされた場合にトラップを発生させ、前記追加ロードされたデータ構造に対してデータアクセスされるようにアクセスを変換する追加ロード前データアクセス手段と、前記追加ロード前データアクセス手段による変換によってアクセスが完了した場合に、トラップから復帰させるための追加ロード前データアクセス完了手段とをさらに具備したことを特徴とする。

【0011】また、前記データ変換移動手段によってデータが移動された追加ロードされたデータ構造から、前記実行中のプログラムが使用していたデータ構造に、データを逆変換して移動させるデータ逆変換移動手段と、前記実行中のプログラムに制御が渡った場合に、追加ロードされたプログラムに制御を渡さないように前記実行中のプログラムの一部を変更するプログラム制御移行中止手段と、前記データ構造／プログラム追加ロード手段によって追加ロードされたデータ構造とプログラムをアンロードする追加データ構造／プログラムアンロード手段とをさらに具備したことを特徴とする。

【0012】

【作用】このような構成によれば、プログラムのバグを発見し、バグ修正情報を作成し、コンパイル／リンクして新しい実行モジュール（プログラム）を作成した場合に、バグ修正情報適用前の古い実行モジュールの停止、すなわちシステムの停止を待たずに、実行モジュールの入れ換えが実行される。

【0013】プログラムの入れ換えに伴うデータ構造の変更に対しては、元のデータ構造に対するアクセスがあった場合に、変換後のデータ構造に対するアクセスに変換されるため、実行モジュール（プログラム）と共に、データに関しても入れ換えが実行される。

【0014】また、バグ修正情報が不適切であった場合等には、バグ修正情報適用前の実行モジュールに制御を戻して、少なくとも前の状態よりも悪くなってしまうことを回避するが、その際にもシステム停止を伴うことが不要とされる。

【0015】

【実施例】以下、図面を参照して本発明の一実施例を説明する。図1は本実施例に係わる計算機システムの概略構成を示すブロック図である。計算機システム10には、プロセッサ12、メモリ14、及び記憶装置16が設けられている。

【0016】メモリ 14 には、プロセッサ 12 にロードされて実行されるプログラムや各種データ等が格納される。図 1 においては、本発明によるプログラム置換を行なう前には、実行中のプログラム 18 が格納されているものとする。実行中のプログラム 18 は、命令列からなるプログラム 20 と、命令列によって処理されるデータ構造 22 からなるものとする。また、プログラム置換の実行時には、追加ロードされたプログラムとデータ構造の一部 24（追加ロードされたプログラム 26、及び追加ロードされたデータ構造 28 からなる）が格納される。

【0017】また、メモリ 14 には、本発明によるプログラム置換機能を実現するプログラム置換プログラムが、例えば OS（オペレーティングシステム）の一部として格納されている。プログラム置換機能については後述する。また、メモリ 14 には、プログラム置換に伴って、実行中のプログラムの一部（プログラム置換前の元の命令）が格納される。

【0018】記憶装置 16 は、例えばハードディスク装置等によって構成されるもので、メモリ 14 中に格納された実行中のプログラム 18 に対応する、ソースプログラム上で更新されコンパイル／リンクされたプログラム 18a 等が格納されているものとする。

【0019】図 2 は、プログラム置換プログラムによって実現されるプログラム置換機能の構成を示すブロック図である。図 2 に示すように、プログラム置換機能は、データ構造／プログラム追加ロード手段 30、データ変換移動手段 32、プログラム制御移行手段 36、追加データ構造／プログラムアンロード手段 38、データ逆変換移動手段 40、プログラム制御移行中止手段 42、プログラム実行一時停止手段 44、プログラム実行再開手段 46、追加ロード前データアクセス手段 48、及び追加ロード前データアクセス完了手段 52 によって構成される。

【0020】データ構造／プログラム追加ロード手段 30、データ変換移動手段 32、及びプログラム制御移行手段 36 は、実行中のプログラムを置換する際に機能するプログラム置換部 54 を構成する。

【0021】また、追加データ構造／プログラムアンロード手段 38、データ逆変換移動手段 40、及びプログラム制御移行中止手段 42 は、プログラム置換部 54 の各機能によって置換されたプログラムを元の状態に戻すプログラム逆置換部 56 を構成する。

【0022】データ構造／プログラム追加ロード手段 30 は、実行中のプログラム 18 に対して置換すべきプログラム 18a をメモリ 14 にロード（追加ロードされたプログラム 26、及び追加ロードされたデータ構造 28）するものである。

【0023】データ変換移動手段 32 は、プログラム置換を実行する際に、実行中のプログラム 18 のデータ構

造 22 のデータを、データ構造／プログラム追加ロード手段 30 によってメモリ 14 に追加ロードされたデータ構造 28 に変換して書き込むと共に、実行中のプログラムのデータ構造 28 に関してはアクセス禁止状態にするものである。データ変換移動手段 32 は、データ構造を変換する際に変換規則に関する情報（変換規則情報 34）を参照し、また変換規則情報 34 に基づいて変換して書き込んだデータが存在する位置に関する情報（移動規則情報 50）を作成する。なお、移動規則情報 50 は、追加ロード前データアクセス手段 48 によってアクセス変換を行なう際に参照されるものである。

【0024】プログラム制御移行手段 36 は、実行中のプログラム 18 中の置換されるべきプログラムの命令が実行されず、データ構造／プログラム追加ロード手段 30 によってメモリ 14 に追加ロードされたプログラムが実行されるように、プログラムの命令の一部を変更するものである。

【0025】追加データ構造／プログラムアンロード手段 38 は、プログラム置換のためにデータ構造／プログラム追加ロード手段 30 によって追加ロードされたデータ構造とプログラムの一部 24 を、例えば誤りが含まれていたために追加ロードされたプログラム／データ構造の実行が中止された際にアンロードするものである。

【0026】データ逆変換移動手段 40 は、置換されたプログラムを元のプログラムに戻す場合に、追加ロードされたデータ構造 28 中のデータを読み込み、元々のデータ構造 22 に逆変換して書き込むと共に、元のデータ構造 22 がアクセスできるようにアクセス許可状態にするものである。

【0027】プログラム制御移行中止手段 42 は、プログラム置換の際にプログラム制御移行手段 36 によって変更された命令を置換前の元の命令に逆変換し、プログラム置換前のプログラムが実行されるようにするものである。

【0028】プログラム実行一時停止手段 44 は、プログラム置換部 54 におけるプログラム置換を行なう際、及びプログラム逆置換部 56 によってプログラムの逆置換を行なう際に、プロセッサ 12 による現在実行中のプログラム実行を一時停止させるものである。

【0029】プログラム実行再開手段 46 は、プログラム実行一時停止手段 44 によって現在実行中のプログラム実行が一時停止されている間に、プログラム置換部 54 におけるプログラム置換、あるいはプログラム逆置換部 56 によってプログラムの逆置換が完了した後、実行中のプログラムの実行を再開させるものである。

【0030】追加ロード前データアクセス手段 48 は、アクセスが禁止されたデータにアクセスする場合に発生されたトラップによって起動され、アクセス禁止されたデータ構造へのアクセスを、追加ロードされ置換されたデータ構造へのアクセスに変換するものである。

【0031】追加ロード前データアクセス完了手段52は、アクセスが禁止されたデータにアクセスする場合に発生されたトラップを、追加ロード前データアクセス手段48による変換によってアクセスを終えた後で、トラップから復帰して通常の処理を再開させるものである。

【0032】次に、本実施例の動作について、図3及び図4に示すフローチャートを参照しながら説明する。まず、実行中のプログラムの一部を置換する動作について、図3を用いて説明する。

【0033】計算機システム10では、メモリ14上にプログラム18がロードされ、プロセッサ12によって、そのプログラムを実行している。実行中のプログラム18は、命令列からなるプログラムと、命令列によって処理されるデータ構造からなる。ここで、実行中のプログラム18のプログラムの一部20と、データ構造の一部22に、例えばバグが検出されるなどして変更する必要が生じたものとする。

【0034】そこで、変更すべきプログラムの一部20と、データ構造の一部22のみをソースプログラム上で更新し、コンパイル／リンクして得られたプログラム18aが記憶装置16に用意される。

【0035】データ構造／プログラム追加ロード手段30は、記憶装置16に格納された更新されたプログラム18aをメモリ14上に追加ロードする（ステップA）。メモリ14上に追加ロードされたプログラムとデータ構造の一部が24である。追加ロードされたプログラムとデータ構造の一部24は、追加ロードされたプログラム26と、追加ロードされたデータ構造28を含んでいる。

【0036】しかし更新したプログラムとデータ構造の一部24をメモリ14上に追加ロードしただけでは、実行中のプログラム18自体は何の影響も受けてなく、プログラムの変更を行なうという事に関しては意味が無い。そこで、以下のようなプログラム置換の処理を実行する。

【0037】まず、プログラム実行一時停止手段44は、実行中のプログラム18の実行を一時停止させる（ステップA2）。実行中のプログラム18が停止されると、データ変換移動手段32は、実行中のプログラムのデータ構造22中のデータを読み込み、変換規則情報34に基づいて、変換後のデータ構造に対応するデータに変換し、追加ロードされたデータ構造28中に書き込む（ステップA3）。そして、実行中のプログラムのデータ構造22に関してはアクセスができないようにアクセス禁止状態にする。なお、データ構造の変換規則情報34は、変更すべきプログラムの一部20と、データ構造の一部22のみをソースプログラム上で更新し、コンパイル／リンクした際に作成され、プログラム置換を開始する前に、メモリ14に格納される。変換規則情報34の詳細については後述する。

【0038】また、プログラム制御移行手段36は、実行中のプログラム20のプログラム置換をすべき位置を示すエントリポイントの命令を、追加ロードされたプログラム26のエントリポイントに無条件分岐する命令に置き換える（ステップA4）。また、プログラム制御移行手段36は、無条件分岐する命令に置換される前の元の命令を保存しておく。

【0039】これによって、実行中のプログラム18のプログラムEとデータ構造22が変更され、プログラム／データ構造を実行する準備が完了したので、プログラム実行再開手段46は、実行中のプログラム18の実行を再開させる（ステップA5）。

【0040】図5にはプログラム置換の具体例を示している。図5（a）はプログラム更新前の状態、図5（b）はプログラム更新後の状態を示している。更新前のプログラム20は、図5（a）に示すように、「レジスタの値を保存する」という命令を含む処理ルーチンであり、他の呼び出し側からの要求によって実行され、一連の命令の実行が完了すると呼び出し側に戻るものである。更新前のプログラム20のエントリポイントの命令は「レジスタの値を保存する」命令となっている。

【0041】前述したようなプログラム置換を実行すると、図5（b）に示すように、更新前のプログラム20のエントリポイントの「レジスタの値を保存する」という命令は、更新後のプログラム26に無条件分岐する命令に置換される。

【0042】なお、プログラム置換前の命令（「レジスタの値を保存する」）は、予め置換前に保存されている。置換前の命令を保存しておくことにより、後述する更新されたプログラムを元に戻す逆変換の処理を実行する際に、無条件分岐する命令に置換されてしまった更新前のプログラムのエントリポイントの命令を元に戻すことができる。

【0043】追加ロードされたプログラム26に制御が移った場合は、通常通りに処理が行なわれる。また、追加ロードされた更新後のプログラム26によって、更新前のプログラム20に制御が渡った場合は、エントリポイントの命令が、追加ロードされたプログラム26のエントリポイントへの無条件分岐命令に置き換えられているので、プログラム20は実行されず、追加ロードされたプログラム26に制御が渡る。

【0044】これによって、プログラム20に代えてプログラム26による通常通りの処理が実行される。また、追加ロードされたプログラム26は、追加ロードされたデータ構造28にアクセスする場合は、通常通りに処理を実行する。

【0045】ところで、元の実行中のプログラム18から、置換されたデータ構造22にアクセスする場合は、データ構造22へのアクセスがデータ変換移動手段32によって禁止されているのでトラップが発生する。

【0046】このトラップによって追加ロード前データアクセス手段48が呼び出される。追加ロード前データアクセス手段48は、元のデータ構造22へのアクセスを、移動規則情報50に基づいて、追加ロードされたデータ構造28へのアクセスに変換する。そして、アクセスを終えた後で、追加ロード前データアクセス完了手段52は、トラップから復帰させて、通常通りの処理を再開させる。なお、移動規則情報50は、データ変換移動手段32が追加ロードされたデータ構造28にデータを変換して書き込む際に作成され、メモリ14に格納される。移動規則情報50及びアクセスの変換の詳細については後述する。

【0047】次に、置換したプログラムの実行を中止して、更新されたプログラムを元に戻す逆変換の動作について、図4に示すフローチャートを参照しながら説明する。例えば、追加ロードしたプログラム26またはデータ構造28に誤りが発見された場合等では、プログラム置換を実行する前の状態に戻すことで、少なくとも元のプログラムにおけるバグの影響を越える障害等の発生を回避させることができる。

【0048】追加ロードされたプログラム26／データ構造28の実行を中止したい場合は、まず、プログラム実行一時停止手段44によって実行中のプログラム18（追加ロードされたデータ構造とプログラムの一部24を含む）の実行を一時停止させる（ステップB1）。

【0049】実行中のプログラム18が停止されると、データ逆変換移動手段40は、追加ロードされたデータ構造28中のデータを読み込み、変換規則情報34に基づいて、元のデータ構造に対応するデータに変換し、元のデータ構造22に逆変換して書き込む（ステップB2）。そして、元のデータ構造22がアクセスができるようにアクセス許可状態にする。

【0050】また、プログラム制御移行中止手段42は、実行中のプログラム20のエントリポイントの命令を、無条件分岐する命令で置き換える前に保存しておいて元の命令に置き換える（ステップB3）。

【0051】これによって、実行中のプログラム18のプログラム20とデータ構造22が元の状態に戻ったので、プログラム実行再開手段46は、実行中のプログラム18の実行を再開させる（ステップB4）。

【0052】追加データ構造／プログラムアンロード手段38は、メモリ14に追加ロードしてあった更新後のデータ構造／プログラムを、記憶装置16にアンロードする（ステップB5）。

【0053】次に、データ構造の変換規則情報の詳細について説明する。変換規則情報は、プログラム置換に伴う更新前のデータ構造と更新後のデータ構造との対応関係を定義するもので、各データ毎に、データタイプとそれらのデータが存在する位置を示すデータ構造の先頭からのオフセットとの組が対応づけられて格納されてい

る。

【0054】図6はデータ構造の変換規則の1番目の例を示す図である。更新前のデータ構造Aは、図6（a）に示すように、long integer型のA1、short integer型のA2、short integer型のA3から構成されているものとする。なお、図6及び図7に示す例では、例えばlong integer型が4バイト、short integer型が2バイトであるものとする。

10 【0055】また、更新後のデータ構造aは、図6（b）に示すように、long integer型のa1、long integer型のa2、long integer型のa3から構成されているものとする。

【0056】この時、変換規則情報Aaは、図6（c）に示すような関係で定義される。すなわち、データ構造Aのオフセット0からのlong integer型の変数は、データ構造aのオフセット0からのlong integer型の変数に対応し、同様にデータ構造Aのオフセット4からのshort integer型の変数はデータ構造aのオフセット4からのlong integer型の変数に対応し、データ構造Aのオフセット6からのshort integer型の変数はデータ構造aのオフセット8からのlong integer型の変数に対応している。

20 【0057】図7はデータ構造の変換規則の2番目の例を示す図である。更新前のデータ構造Bは、図7（a）に示すように、long integer型のB1、long integer型のB2から構成されているものとする。

30 【0058】また、更新後のデータ構造bは、図7（b）に示すように、long integer型のb1、long integer型のb2、long integer型のb3から構成されているものとする。

【0059】この時、変換規則Bbは、図7（c）に示すような関係で定義される。すなわち、データ構造Bのオフセット0からのlong integer型の変数はデータ構造bのオフセット0からのlong integer型の変数に対応し、データ構造Bのオフセット4からのlong integer型の変数はデータ構造bのオフセット4からのlong integer型の変数に対応し、データ構造bのオフセット8からのlong integer型の変数は新たに追加されている。

40 【0060】データ変換移動手段32は、図6（c）または図7（c）に示すような変換規則情報に基づいて、実行中のプログラム18のデータ構造22のデータを、追加ロードされたデータ構造28に合わせて変換して書き込むことができる。

50 【0061】また、データ逆変換移動手段40は、同様にして変換規則情報に基づいて、追加ロードされたデー

タ構造 28 のデータを、元のデータ構造 22 に合わせて変換して書き込むことができるので、プログラム置換の前の元の状態にすることができる。

【0062】なお、図 6 及び図 7 では、integer 型のデータ型についてのみ示しているが、他のデータ型に応用することも可能である。次に、移動規則情報、及び追加ロードされたデータ構造へのアクセス（アクセスの変換）について詳細を説明する。移動規則情報は、プログラム置換に伴う更新前のデータ構造と更新後のデータ構造のそれぞれにおけるデータの位置の対応関係を定義するもので、データが位置する開始番地とそれらのデータ長との組が対応づけられて格納されている。

【0063】図 8 は追加ロードされたデータ構造へのアクセスの例を示す図である。なお、図 8 ではページ単位でアクセス制御されるものとする。また、アクセス制御は、例えばページ毎に設けられた保護ビットに“1”をセットすることにより、アクセス禁止にする方式がとられる。

【0064】図 8 (a) は更新前のデータ構造（データ構造 22）を含むページ P で、プログラム置換に伴ってアクセス禁止状態にされているものとする。このページ P の中には X 番地に long integer 型の変数からなるデータ構造 A が存在し、Y 番地に 2 つの short integer 型の変数からなるデータ構造 B が存在している。

【0065】図 8 (b) は更新後のデータ構造（データ構造 28）を含むページ Q で、アクセス可能状態になっている。このページ Q の中には Z 番地に 2 つの long integer 型の変数からなるデータ構造 b が存在している。データ構造 b は、図 8 (a) 中のデータ構造 B を更新したものである。

【0066】ページ P はアクセス禁止状態になっているので、このページ上に存在する全てのデータ構造へのアクセスにおいてトラップが発生する。トラップが発生した場合は、追加ロード前データアクセス手段 48 が呼び出される。追加ロード前データアクセス手段 48 は、トラップ処理の中で、図 8 (c) に示すような移動規則情報 R を検査する。

【0067】追加ロード前データアクセス手段 48 は、移動規則情報 R を参照して、トラップが発生した場合に、そのページ上のデータ構造にアクセスすべきか、あるいはデータ構造が更新されているため他のデータ構造にアクセスすべきかを判定することができる。

【0068】例えば、ページ P の X 番地のデータ構造 A にアクセスしてトラップが発生した場合、X 番地からのデータ構造 A が移動規則 R に登録されていないので、追加ロード前データアクセス手段 48 は、ページ P 上の X 番地のデータ構造 A にアクセスさせる。

【0069】一方、Y 番地のデータ構造 B にアクセスしてトラップが発生した場合、Y 番地からのデータ構造 B

が移動規則 R に登録されているので、追加ロード前データアクセス手段 48 は、ページ P 上の Y 番地のデータ構造 B の代わりに、ページ Q 上に Z 番地のデータ構造 b にアクセスさせる。

【0070】こうして、移動規則情報に応じてアクセスを変換するので、プログラム置換を行なう際に、追加ロードされたデータ構造 28 に対しても正しくアクセスすることができる。

【0071】このようにして、実行中のプログラム 18 にバグが検出された場合に、バグ修正情報を作成し、このバグ修正情報が適用されたプログラムを改めてコンパイル／リンクして新しい実行モジュールを作成した場合に、実行中のプログラムが停止されるシステム停止の機会を待つことなく、適宜、プログラム置換機能による追加ロードされたプログラム 26 及びデータ構造 28 の置換によって修正することができる。

【0072】追加ロードされたデータ構造 28 に対しては、追加ロードされる前の元のデータ構造 22 に対するアクセスがあった場合にトラップを発生させて、トラップ処理のなかでアクセス変換を行なうことによって、アクセスできるように制御される。

【0073】また、追加ロードしたプログラム 26 またはデータ構造 28 に誤りが発見された場合でも、システム停止の機会を待つことなく、バグ修正情報適用前の状態に制御を戻すことができる。

#### 【0074】

【発明の効果】以上詳述したように本発明によれば、システムを停止させることなく、実行中のプログラムの部分的な置換を行なうことが可能となるものである。

#### 【図面の簡単な説明】

【図 1】本発明の一実施例に係わる計算機システムの概略構成を示すブロック図。

【図 2】本発明によるプログラム置換機能の構成を示すブロック図。

【図 3】本実施例におけるプログラム置換の動作を説明するためのフローチャート。

【図 4】本実施例におけるプログラム逆変換の動作を説明するためのフローチャート。

【図 5】本実施例におけるプログラム置換の具体例を示す図。

【図 6】本実施例におけるデータ構造の変換規則の 1 番目の例を示す図。

【図 7】本実施例におけるデータ構造の変換規則の 2 番目の例を示す図。

【図 8】本実施例における追加ロードされたデータ構造へのアクセス（アクセスの変換）を説明するための図。

#### 【符号の説明】

10…計算機システム、12…プロセッサ、14…メモリ、16…記憶装置、18…実行中のプログラム、20、26…プログラム、22、28…データ構造、24

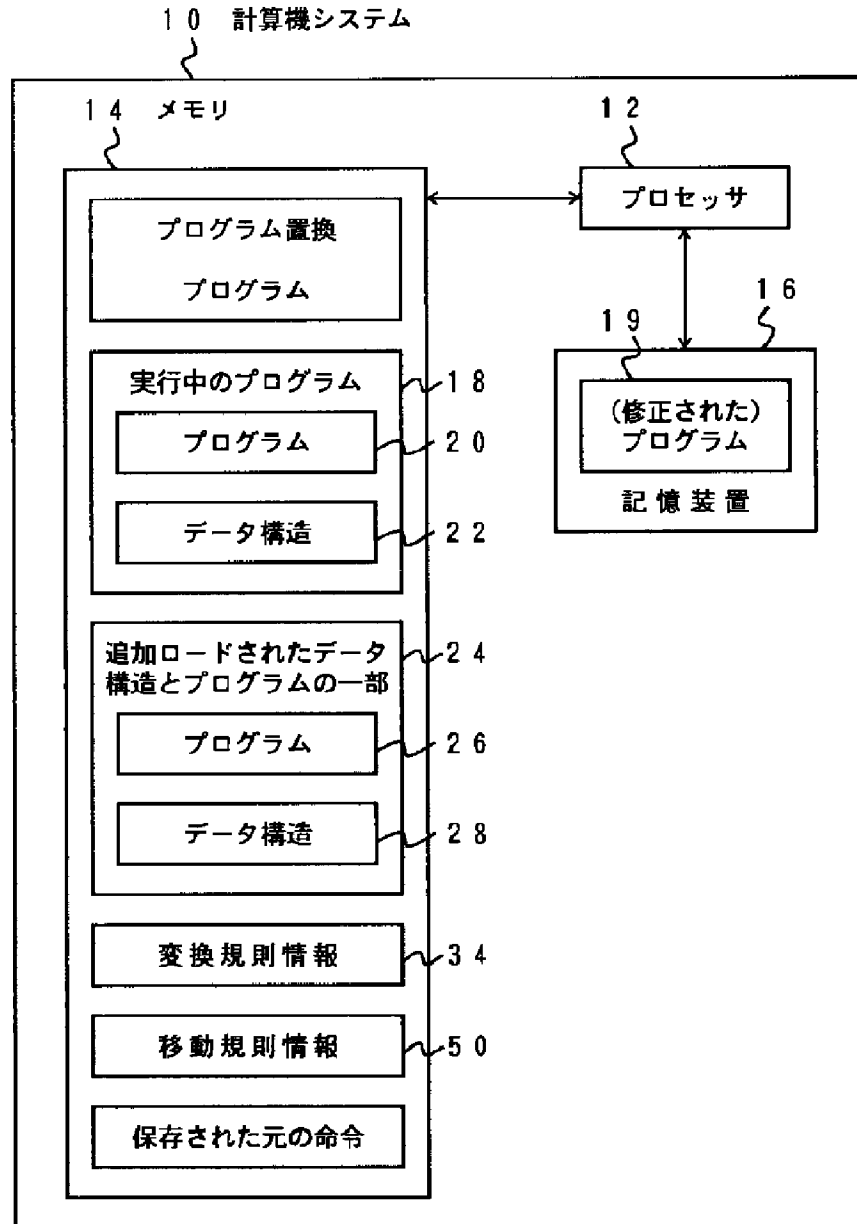
13

…追加ロードされたプログラムとデータ構造の一部、30…データ構造／プログラム追加ロード手段、32…データ変換移動手段、34…変換規則情報、36…プログラム制御移行手段、38…追加データ構造／プログラムアンロード手段、40…データ逆変換移動手段、42…

14

プログラム制御移行中止手段、44…プログラム実行一時停止手段、46…プログラム実行再開手段、48…追加ロード前データアクセス手段、50…移動規則情報、52…追加ロード前データアクセス完了手段、54…プログラム置換部、56…プログラム逆置換部。

【図1】





修正されたプログラム

プログラム逆置機部

38

56

追加データ構造 / プログラムアンロード手段

アンロード

40

データ逆変換移動手段

データ読み込み

42

プログラム制御移行中止手段

変換規則情報

移動規則情報

保存された元の命令

50

34

28

26

24

22

20

プログラム

データ構造

追加ロードされたデータ構造とプログラムの一部

プログラム

データ構造

18

14 メモリ

実行中のプログラム

ロード

分岐先変更

アクセス / 禁止

32

データ書き込み

データ変換移動手段

データ読み込み

追加ロード手段

データ構造 / プログラム追加ロード手段

30

54

修正されたプログラム

プログラム置換機部

44

プログラム実行一時停止手段

プログラム実行再開手段

46

48

追加ロード前データアクセス手段

追加ロード前データアクセス手段

52

追加ロード前データアクセス完了手段

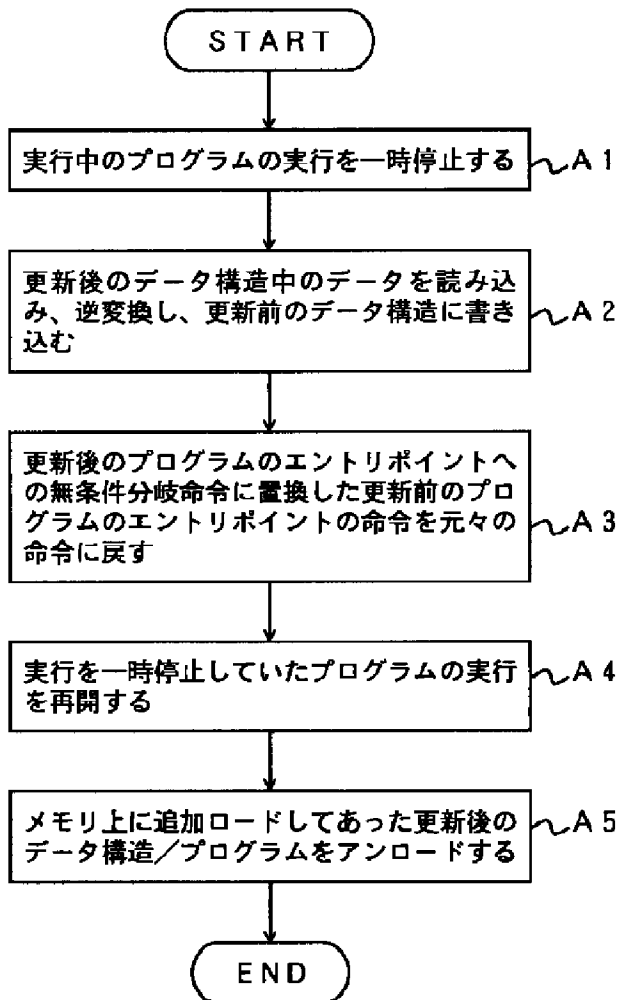
トラップ発生

アクセスされた変換されたアクセス

アクセス完了

トラップ復帰

【図 3】



【図 6】

(a) 更新前のデータ構造 A

A 1 (long)	
A 2 (short)	A 3 (short)

(b) 更新後のデータ構造 a

a 1 (long)
a 2 (long)
a 3 (long)

(c) 変換規則情報 A a

データ構造 A		データ構造 a	
オフセット	タイプ	オフセット	タイプ
+ 0	long	+ 0	long
+ 4	short	+ 4	long
+ 6	short	+ 8	long

【図 7】

(a) 更新前のデータ構造 B

B 1 (long)
B 2 (long)

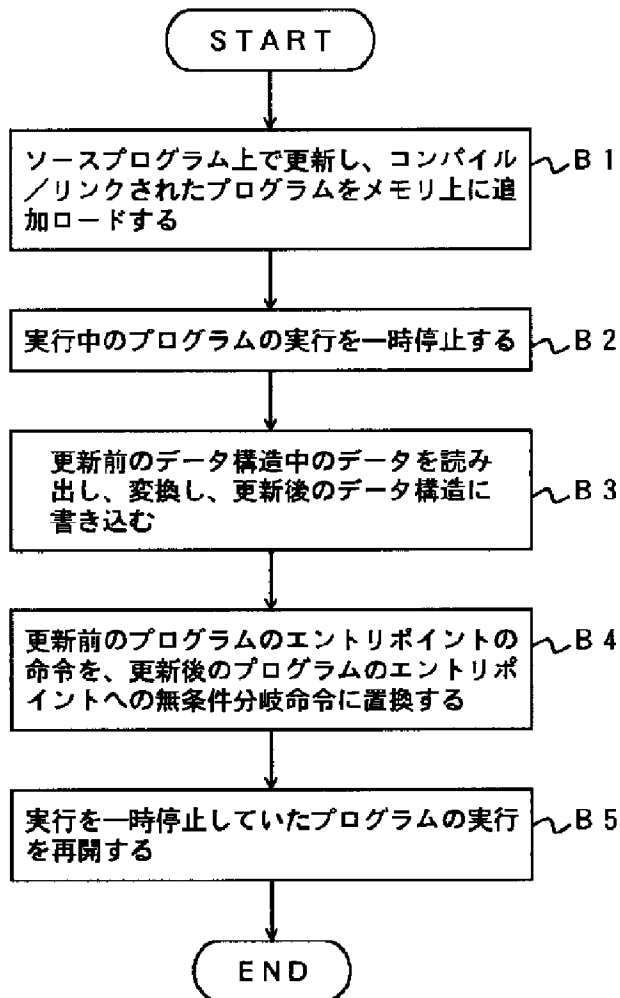
(b) 更新後のデータ構造 b

b 1 (long)
b 2 (long)
b 3 (long)

(c) 変換規則情報

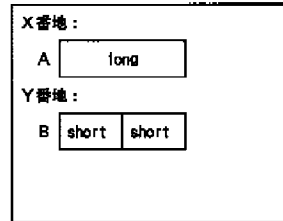
データ構造 B		データ構造 b	
オフセット	タイプ	オフセット	タイプ
+ 0	long	+ 0	long
+ 4	long	+ 4	long
-	-	+ 8	long

【図4】

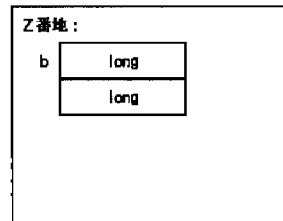


【図8】

(a) 更新前のデータ構造を含むページP (アクセス禁止)



(b) 更新後のデータ構造を含む追加ロードされたページQ (アクセス許可)



(c) 移動規則情報

更新前		更新後	
開始番地	長さ	開始番地	長さ
Y	4	Z	8
⋮	⋮	⋮	⋮

【図5】

